Algorithm 1 Stable Monotonic Chunkwise Attention Decoding

```
Input: encoder features H = \{h_1, \dots, h_U\}, output index
      i, decoder hidden state s_i, output label y_i, endpoint t_i,
       sigmoid function \sigma(\cdot), attention chunk width w
  1: Initialize s_0 = \vec{0}, \ y_0 = \langle sos \rangle, \ t_0 = 1, \ i = 1
      while y_{i-1} \neq \langle eos \rangle do
            \begin{aligned} \text{for } j &= t_{i-1} \text{ to } U \text{ do} \\ e_{i,j} &= g \frac{v_m^\top}{\|v_m\|} \text{tanh}(W_m^s s_{i-1} + W_m^h h_j + b_m) + r \end{aligned}
  3:
  4:
                  p_{i,j} = \sigma(e_{i,j})
  5:
                  if p_{i,j} \geq 0.5 then
  6:
                        \quad \mathbf{for}\; k = j - w + 1\; \mathbf{to}\; j\; \mathbf{do}
  7:
                              u_{i,k} = v_c^{\top} \tanh(W_c^s s_{i-1} + W_c^h h_k + b_c)
  8:
                       cia iof c_i = \sum_{k=j-w+1}^j \frac{\exp(u_{i,k})}{\sum_{l=j-w+1}^j \exp(u_{i,l})} h_k t_i = j
  9:
10:
11:
                        break
12:
                  end if
13:
            end for
14:
            if p_{i,j} < 0.5, \ \forall j \in \{t_{i-1}, \cdots, U\} then
15:
                  c_i = \vec{0}, \quad t_i = t_{i-1}
16:
17:
            y_i \sim \text{Decoder}(s_{i-1}, y_{i-1}, c_i), \quad i = i + 1
18:
19: end while
```

Algorithm 2 Stable Monotonic Chunkwise Attention Training

```
Input: encoder features H = \{h_1, \dots, h_U\}, output index i,
        decoder hidden state s_i, output label y_i, sigmoid function
        \sigma(\cdot), attention chunk width w, Gaussian noise \epsilon
   1: s_0 = \vec{0}, y_0 = \langle sos \rangle, \alpha_{0,0} = 1, \alpha_{0,k} = 0 (k \neq 0), i = 1
       while y_{i-1} \neq \langle eos \rangle do
               for j = 1 to U do
                     e_{i,j} = g \frac{v_m^{\top}}{\|v_m\|} \tanh(W_m^s s_{i-1} + W_m^h h_j + b_m) + r
p_{i,j} = \sigma(\text{Energy}(s_{i-1}, h_j) + \epsilon)
                      \alpha_{i,j} = p_{i,j} \prod_{k=1}^{j-1} (1 - p_{i,k})
  6:
               end for
  7:
               for j = 1 to U do
                     \begin{aligned} & J = 1 \text{ to 5 Co} \\ & u_{i,j} = v_c^{\top} \tanh(W_c^s s_{i-1} + W_c^h h_j + b_c) \\ & \beta_{i,j} = \sum_{k=j}^{j+w-1} \frac{\alpha_{i,k} \exp(u_{i,j})}{\sum_{l=k-w+1}^k \exp(u_{i,l})} \end{aligned}
 10:
              end for c_i = \sum_{j=1}^{U} \beta_{i,j} h_j \\ y_i \sim \operatorname{Decoder}(s_{i-1}, y_{i-1}, c_i), \quad i = i+1
11:
12:
 13:
14: end while
```

Algorithm 3 Monotonic Truncated Attention Decoding

```
Input: encoder features H = \{h_1, \dots, h_U\}, output index
      i, decoder hidden state s_i, output label y_i, endpoint t_i,
      sigmoid function \sigma(\cdot), attention chunk width w
  1: Initialize s_0 = \vec{0}, \ y_0 = \langle sos \rangle, \ t_0 = 1, \ i = 1
      while y_{i-1} \neq \langle eos \rangle do
            for j = 0 to U_{\perp} do
                e_{i,j} = g \frac{v_m^\top}{||v_m||} \tanh(W_m^s s_{i-1} + W_m^h h_j + b_m) + r
p_{i,j} = \sigma(e_{i,j})
  3:
  4:
  5:
                 \alpha_{i,j} = p_{i,j} \prod_{k=1}^{j-1} (1-p_{i,k}) if p_{i,j} \geq 0.5 then
  6:
  7:
                       c_i = \sum_{k=1}^j \alpha_{i,k} h_k
t_i = j
  8:
  9:
 10:
                 end if
 11:
            end for
 12:
            if p_{i,j} < 0.5, \ \forall j \in \{t_{i-1}, \cdot \cdot \cdot, U\} then
 13:
                 c_i = \vec{0}, \quad t_i = t_{i-1}
 14:
 15:
            y_i \sim \text{Decoder}(s_{i-1}, y_{i-1}, c_i), \quad i = i+1
 16:
17: end while
```

Algorithm 4 Monotonic Truncated Attention Training

```
Input: encoder features H=\{h_1,\cdots,h_U\}, output index i, decoder hidden state s_i, output label y_i, sigmoid function \sigma(\cdot), attention chunk width w, Gaussian noise \epsilon

1: s_0=\vec{0},\ y_0=\langle sos\rangle,\ \alpha_{0,0}=1,\ \alpha_{0,k}=0 (k\neq 0),\ i=1

2: while y_{i-1}\neq \langle eos\rangle do

3: for j=1 to U do

4: e_{i,j}=g\frac{v_m^v}{\|v_m\|} \mathrm{tanh}(W_m^s s_{i-1}+W_m^h h_j+b_m)+r

5: p_{i,j}=\sigma(\mathrm{Energy}(s_{i-1},h_j)+\epsilon)

6: \alpha_{i,j}=p_{i,j}\prod_{k=1}^{j-1}(1-p_{i,k})

7: end for

8: c_i=\sum_{j=1}^U\alpha_{i,j}h_j

9: y_i\sim\mathrm{Decoder}(s_{i-1},y_{i-1},c_i),\ i=i+1

10: end while
```